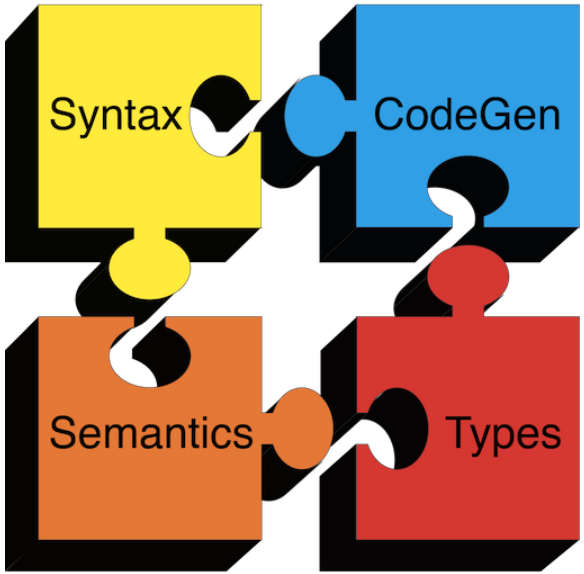

Compilers

Introduction to Code Generation

Alex Aiken

Code Generation

- We focus on generating code for a stack machine with accumulator
- We want to run the resulting code on a real machine
  - e.g., the MIPS processor (or simulator)
- We simulate stack machine instructions using MIPS instructions and registers

Code Generation

- The accumulator is kept in MIPS register $a0
- The stack is kept in memory
  - The stack grows towards lower addresses
  - Standard convention on MIPS
- The address of the next location on the stack is kept in MIPS register $sp
  - The top of the stack is at address $sp + 4

Code Generation

MIPS architecture

- Prototypical Reduced Instruction Set Computer (RISC)
- Most operations use registers for operands & results
- Use load & store instructions to use values in memory
- 32 general purpose registers (32 bits each)
  - We use $sp, $a0 and $t1 (a temporary register)

- Read the SPIM documentation for details

Code Generation

- lw reg_1 offset(reg_2)
  - Load 32-bit word from address reg_2 + offset into reg_1
- add reg_1 reg_2 reg_3
  - $reg_1 \leftarrow reg_2 + reg_3$
- sw reg_1 offset(reg_2)
  - Store 32-bit word in reg_1 at address reg_2 + offset
- addiu reg_1 reg_2 imm
  - $reg_1 \leftarrow reg_2$ + imm
  - "u" means overflow is not checked
- li reg imm
  - reg $\leftarrow$ imm

The stack-machine code for 7 + 5 in MIPS:

```
acc ← 7
push acc

acc ← 5
acc ← acc + top_of_stack

pop
```

```
li $a0 7
sw $a0 0($sp)
addiu $sp $sp -4
li $a0 5
lw $t1 4($sp)
add $a0 $a0 $t1
addiu $sp $sp 4
```